# *Exploring models*

## *Summary, explainability, and prediction*

R.W. Oldford

UNIVERSITY OF WATERLOO

## *Modelling*

Recall how J.W. Tukey and M.B. Wilk (1966) likened analyzing data to conducting experiments.

[1] Emphasis added in bold.

# *Modelling*

Recall how J.W. Tukey and M.B. Wilk (1966) likened analyzing data to conducting experiments.

They suggested that objectives in data analysis that are comparable to those of experimentation are[1]

1. "to achieve more specific description of what is loosely known or suspected;
2. "to find unanticipated aspects in the data, and **to suggest unthought-of models** for the data's **summarization and exposure**;
3. "to employ the data **to assess** the (always incomplete) **adequacy** of a **contemplated model**;
4. "to provide both incentives and guidance for further analysis of the data; and
5. "to keep the investigator usefully stimulated while he absorbs the feeling of his data and considers what to do next."

Here, model is generally (though not exclusively) to be understood in a more formal mathematical sense. Tukey and Wilk also cautioned against taking them too seriously. In their words, "**Models must be used but must never be believed.**"

[1] Emphasis added in bold.

# *Models – what are they for?*

Models serve several purposes. Here are some:

- to provide a summary pattern of the data, that is those models
  - contemplated to hold from past experience or prior knowledge
  - or those only just discovered empirically from the data
- to help explain how such patterns might have come about
  - competing models can equally well summarize known or discovered patterns
  - amongst these, we typically prefer models which are interpretable
  - amongst these, simple is preferred over complexity
- to predict as yet unobserved values of some variates given the values observed on others

George Box popularized the scientific sense of models with his oft-quoted phrase

"All models are wrong, but some are useful."

A less well known quote from Box is:

> *Since all models are wrong the scientist must be alert to what is importantly wrong. It is inappropriate to be concerned about mice when there are tigers abroad.*

G.E.P. Box (1976) "Science and Statistics", *Journal of the American Statistical Association*, *71*, pp. 791-799.

UNIVERSITY OF WATERLOO

# *Response models*

These are the most common statistical models and come in huge variety of forms.

Variates are distinguished by those which are response variates (the $y$s) and those which are explanatory (the $x$s). In general, each of $x$ and $y$could be multivariate (e.g. $p$ explanatory variates $x_1, \ldots, x_p$; $m$ response variates $y_1, \ldots, y_m$).

**Most commonly**:

- the response is univariate $y$ (i.e. $m = 1$) and modelled as a univariate random variable $Y$;
- the explanatory variate values are taken as given (either fixed by design, or conditioned on by choice) and
- we try to model the conditional expectation

  $$E(Y \mid x_1, \ldots, x_p) = \mu(x_1, \ldots, x_p)$$

  as a function $\mu()$ of the explanatory variates $x_1, \ldots, x_p$.
- we **fit** the model using observed values of all variates, giving the **estimate** $\widehat{\mu}(x_1, \ldots, x_p)$ of the **estimand** $\mu(x_1, \ldots, x_p)$,
- we make **inferences** from the model about $\mu()$ using the **estimator** $\widetilde{\mu}(x_1, \ldots, x_p)$ and its distribution.
- when $\mu()$ is expressed in terms of a finite number of unknown parameters, say $\theta_1, \ldots, \theta_k$, we say that it is a **parametric model** with parameter estimates $\widehat{\theta}_1, \ldots, \widehat{\theta}_k$ and corresponding estimators $\widetilde{\theta}_1, \ldots, \widetilde{\theta}_k$.

# *Response models - examples*

**Regression models**

$$\begin{aligned} Y &= \mu(x_1, \ldots, x_q) + R \\ E(R) &= 0 \\ R &\sim F_R(r; \sigma) \end{aligned}$$

with normal regression models having $F_R$ be a normal or Gaussian distribution ($R \sim G(0, \sigma)$).

This is also rewritten as

$$\begin{aligned} Y \mid x_1, \ldots, x_q &\sim F_Y(y; x_1, \ldots, x_q) \\ E(Y \mid x_1, \ldots, x_q) &= \mu(x_1, \ldots, x_q). \end{aligned}$$

The dependency of the mean on the explanatory variates is then usually modelled. Note that this model is **generative** in that it describes how the response values might have been **generated**.

Such models include the **linear model** whereby

$$\mu(x_1, \ldots, x_q) = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p.$$

Here linear refers to the mean model being linear in the unknown parameters $\theta_i$.

(There are non-linear regression models as well.)

UNIVERSITY OF WATERLOO

## *Response models - examples*

**Generalizing the linear model**

A slight generalization is to instead model a function of the conditional mean, as in the so-called **generalized linear model** where now there is a known function $g(\mu)$ called the **link** function and we model

$$g(\mu(x_1, \ldots, x_q)) = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p$$

with everything else as before.

Another way we might generalize the linear model is to model the mean as

$$\mu(x_1, \ldots, x_q) = \theta + h_1(x_1) + \cdots + h_p(x_p)$$

where $h_i(x_i)$ are arbitrary functions, each of only a single explanatory variate $(x_i)$. This is called an **additive model** (being additive in functions of the explanatory variates).

And, if additionally, it is $g(\mu)$ that is modelled additively, the model is called a **generalized additive model**

These are only a few of the many models that are possible.

## *Response models - in R a consistent interface*

### Formulas

- terms in the formula can be any specified function of one (or more) explanatory variates named in the data set
- in some cases (e.g. for generalized additive models), the function expression `s()` is reserved to indicate a **nonparametric smooth** function to be fitted as the additive term. E.g.

  $$y \; x1 + s(x2) + s(x3)$$

  specifies that the additive term $x1$ enters the model as a usual linear model term, while $s(x2)$ and $s(x3)$ indicate that the model terms for $x2$ and $x3$ are to be separate smooth additive functions for each of $x2$ and $x3$ respectively.
- terms are joined together with binary operators `+`, `-`, `:`, `*`, and `/`, where for terms `a` and `b` we understand that
  - `+ b` indicates adding a separate term `b` to the model,
  - `- b` indicates removing the term `b` from the model,
  - `a:b` indicates an *interaction* term between `a` and `b` be added,
  - `a*b` is a short-hand equivalent to `a + b + a:b`,
  - `a/b` indicates `b` *nested* within `a` and is equivalent to `a + a:b`
- `poly(x, p)` specifies a polynomial in `x` of degree `p` (uses orthogonal polynomials)

# *Response models - in R a consistent interface*

**Fitted models**

Once estimated from the available data, there are common interfaces we expect to have with the fitted model. Suppose the fitted model has been assigned to the variable `myfit`, then common interactions we might expect include

- `summary(myfit)` should return (and print) a **statistical summary** of the data such as
  - an overall measure of the quality of the fit
  - an indication of the statistical significance of each term in the model
- `plot(fit)` should produce one or more plots that summarize the fit and provide some diagnostic tools for assessing its quality
- `predict(fit , ... )` provide predictions of the response (typically its estimated conditional mean) at any collection of variate values
  - requires a data set of new values for every variate named in the model formula
  - often also produces prediction intervals for a new observation and confidence intervals for the conditional mean
- `str(fit)` reveals the structure of the fitted model. Here we expect to also find `myfit$residuals` containing the residuals, or deviations, of the observed responses from their fitted conditional mean

# *Facebook data - fitting linear models*

Linear models are fitted in R using the `lm()` function.

```
fit1 <- lm(log10(Impressions) ~ Paid, data = facebook)
summary(fit1)
```

```
##
## Call:
## lm(formula = log10(Impressions) ~ Paid, data = facebook)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.25955 -0.32022 -0.09619  0.28444  2.03001
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.01543    0.02655 151.236  < 2e-16 ***
## Paid         0.21142    0.05031   4.203 3.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5038 on 497 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.03432,    Adjusted R-squared:  0.03238
## F-statistic: 17.66 on 1 and 497 DF,  p-value: 3.128e-05
```

UNIVERSITY OF WATERLOO

# *Facebook data - contents of linear fits*

## Extracting contents

```r
fit1$coefficients
```

```
## (Intercept)        Paid
##   4.0154262   0.2114186
```

```r
head(model.matrix(fit1))
```

```
##   (Intercept) Paid
## 1           1    0
## 2           1    0
## 3           1    0
## 4           1    1
## 5           1    0
## 6           1    0
```

```r
head(fit1$residuals)
```

```
##          1          2          3          4          5          6
## -0.3086231  0.2646283 -0.3746467  0.7175934  0.1179211  0.3036590
```

```r
# And prediction (based on the estimated mean don't forget)
predict(fit1, newdata = data.frame(Paid = c(0,1)))
```

```
##        1        2
## 4.015426 4.226845
```

```r
# The predicted mean increase in Impressions for paid advertising
diff(10^predict(fit1, newdata = data.frame(Paid = c(0,1))))
```

```
##        2
## 6497.921
```

# *Facebook data - linear model with a factor*

Recall that Category took values Product, Inspiration, Action

```
fit2 <- lm(log10(Impressions) ~ Category, data = facebook)
summary(fit2)
```

```
##
## Call:
## lm(formula = log10(Impressions) ~ Category, data = facebook)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3727 -0.3074 -0.1079  0.2854  1.9168
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          4.12860    0.03482 118.583   <2e-16 ***
## CategoryInspiration -0.09723    0.05379  -1.807   0.0713 .
## CategoryProduct     -0.09449    0.05672  -1.666   0.0963 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5105 on 497 degrees of freedom
## Multiple R-squared:  0.008645,   Adjusted R-squared:  0.004655
## F-statistic: 2.167 on 2 and 497 DF,  p-value: 0.1156
```

# *Facebook data - contents of linear model with a factor*

## Extracting contents

```r
fit2$coefficients
```

```
##         (Intercept) CategoryInspiration     CategoryProduct
##          4.12860385         -0.09722799         -0.09449137
```

```r
head(model.matrix(fit2))
```

```
##   (Intercept) CategoryInspiration CategoryProduct
## 1           1                   0               1
## 2           1                   0               1
## 3           1                   1               0
## 4           1                   0               1
## 5           1                   0               1
## 6           1                   0               1
```

```r
head(fit2$residuals)
```

```
##           1           2           3           4           5           6
## -0.32730938  0.24594205 -0.39059638  0.91032577  0.09923478  0.28497275
```

```r
# And prediction on original scale of Impressions
10^predict(fit2, newdata = data.frame(Category = factor(levels(facebook$Category))))
```

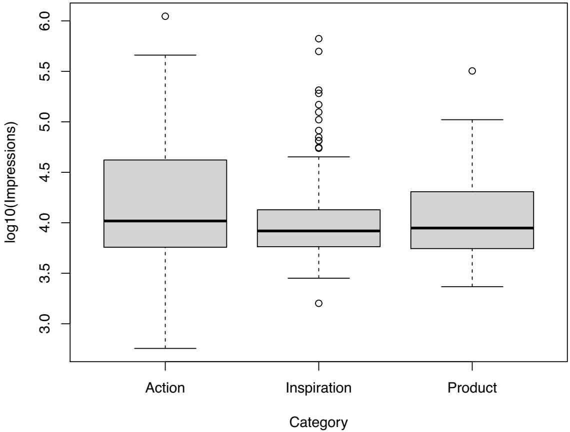
```
##        1        2        3
## 13446.33 10749.19 10817.14
```

Conclusions?

# *Facebook data - other uses of formula*

Formulas are also used by other functions (e.g. `boxplot()`)

```
boxplot(log10(Impressions) ~ Category, data = facebook, col = "lightgrey")
```

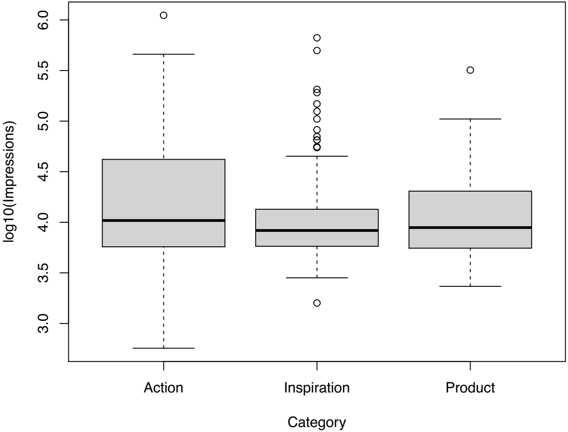



Comments?

# *Facebook data - other uses of formula*

Formulas are also used by other functions (e.g. `boxplot()`)

```
boxplot(log10(Impressions) ~ Category, data = facebook, col = "lightgrey")
```

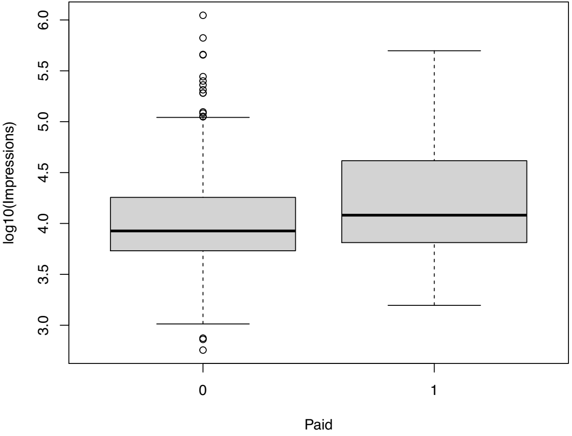



Comments? How is this "model" different from the one constructed by `lm()`'?

# *Facebook data - other uses of formula*

How about `log10(Impressions)` as a function of `Paid`?

```
boxplot(log10(Impressions) ~ Paid, data = facebook, col = "lightgrey")
```

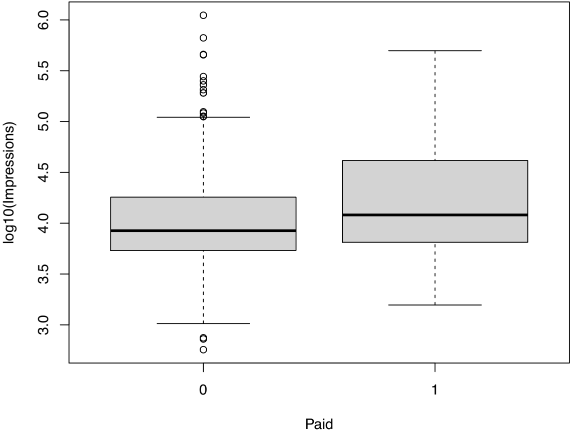



Comments?

# *Facebook data - other uses of formula*

How about `log10(Impressions)` as a function of `Paid`?

```
boxplot(log10(Impressions) ~ Paid, data = facebook, col = "lightgrey")
```

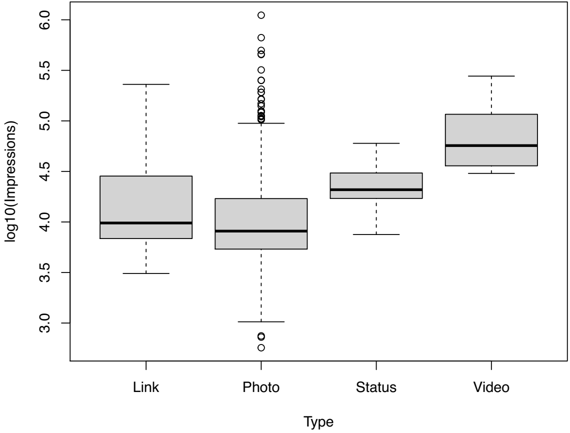



Comments? How is this model formula interpreted?

# *Facebook data - other uses of formula*

How about `log10(Impressions)` as a function of `Type`?

```
boxplot(log10(Impressions) ~ Type, data = facebook, col = "lightgrey")
```

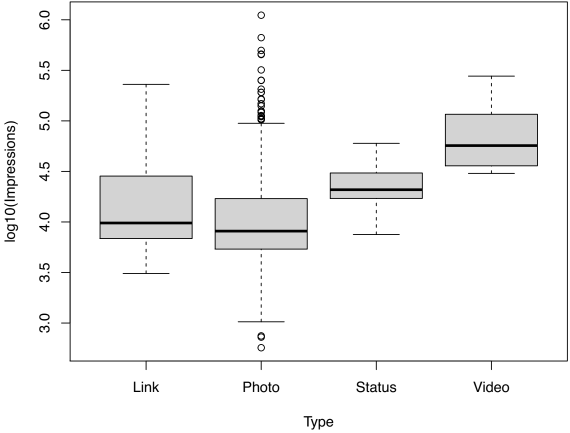



Comments?

# *Facebook data - other uses of formula*

How about `log10(Impressions)` as a function of `Type`?

```
boxplot(log10(Impressions) ~ Type, data = facebook, col = "lightgrey")
```

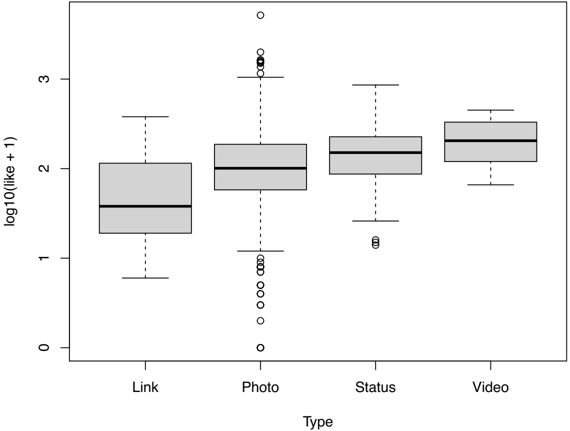



Comments? How is this model formula interpreted?

# *Facebook data - other uses of formula*

How about `log10(Impressions)` as a function of `Type`?

```r
boxplot(log10(like +1) ~ Type, data = facebook, col = "lightgrey")
```

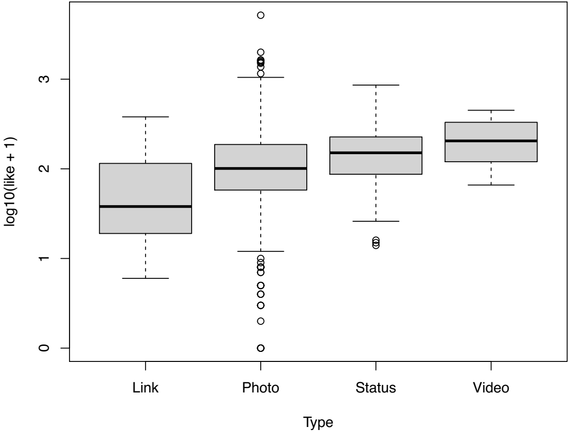



Comments?

# *Facebook data - other uses of formula*

How about `log10(Impressions)` as a function of `Type`?

```
boxplot(log10(like +1) ~ Type, data = facebook, col = "lightgrey")
```

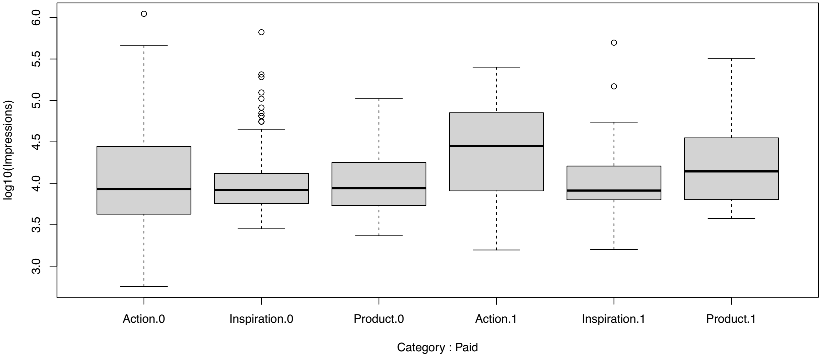



Comments? How is this model formula interpreted?

# *Facebook data - other uses of formula*

This works well when explanatory variates are categorical.

```
boxplot(log10(Impressions) ~ Category + Paid, data = facebook, col = "lightgrey")
```

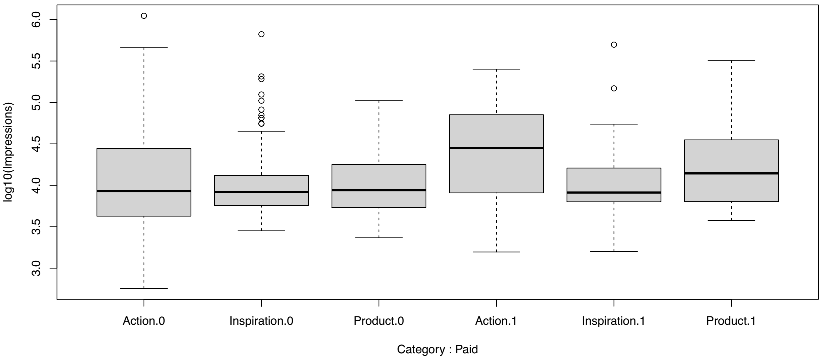



Comments?

# *Facebook data - other uses of formula*

This works well when explanatory variates are categorical.

```
boxplot(log10(Impressions) ~ Category + Paid, data = facebook, col = "lightgrey")
```



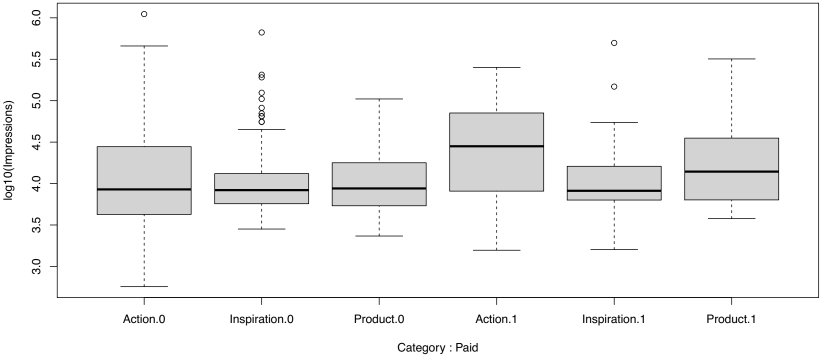

Comments?

Note the labels on the horizontal axis.

# *Facebook data - other uses of formula*

This works well when explanatory variates are categorical.

```
boxplot(log10(Impressions) ~ Category + Paid, data = facebook, col = "lightgrey")
```




Comments?

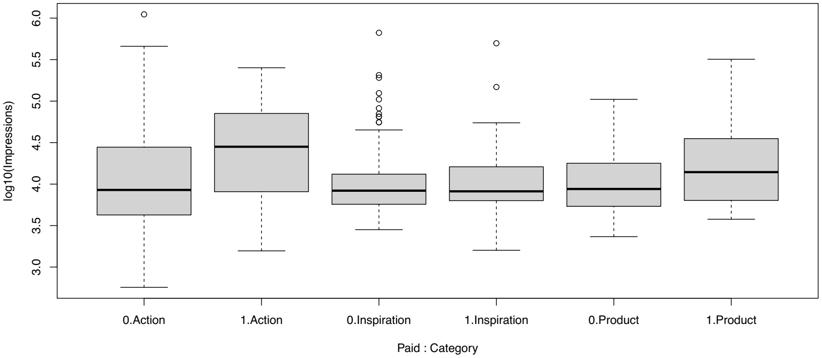
Note the labels on the horizontal axis.How is this model formula interpreted?

UNIVERSITY OF WATERLOO

# *Facebook data - other uses of formula*

What has changed here?

```
boxplot(log10(Impressions) ~ Paid + Category, data = facebook, col = "lightgrey")
```

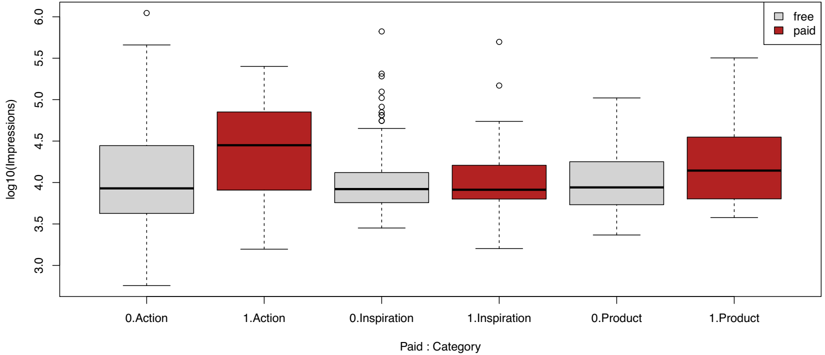



Note the labels on the horizontal axis.

# *Facebook data - other uses of formula*

What has changed here?

```r
boxplot(log10(Impressions) ~ Paid + Category, data = facebook,
            col = rep(c("lightgrey", "firebrick"), 3))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```



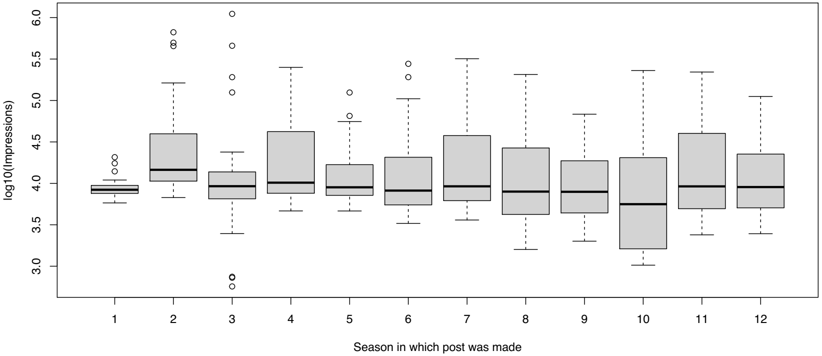

Note the labels on the horizontal axis.

Comments?

# *Facebook data - other uses of formula*

By month

```
boxplot(log10(Impressions) ~ Post.Month,
        xlab = "Season in which post was made",
        data = facebook, col = "lightgrey"
        )
```



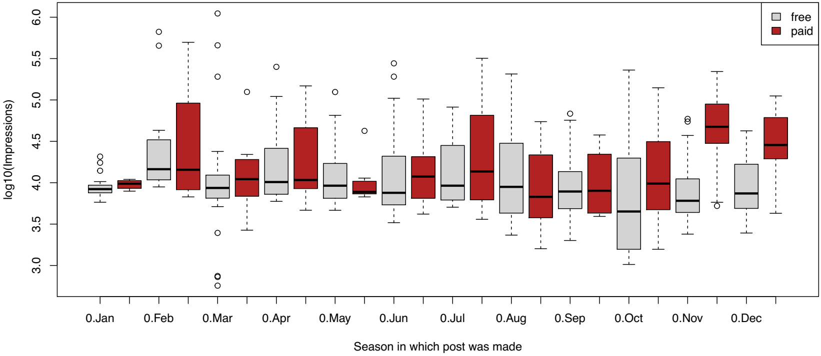

Comments?

UNIVERSITY OF WATERLOO

# *Facebook data - other uses of formula*

How about?

```
boxplot(log10(Impressions) ~ Paid + cut(Post.Month, 4,
                                        labels = c("Winter", "Spring",
                                                   "Summer", "Fall")),
        xlab = "Season in which post was made",
        data = facebook, col = rep(c("lightgrey", "firebrick"), 4))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

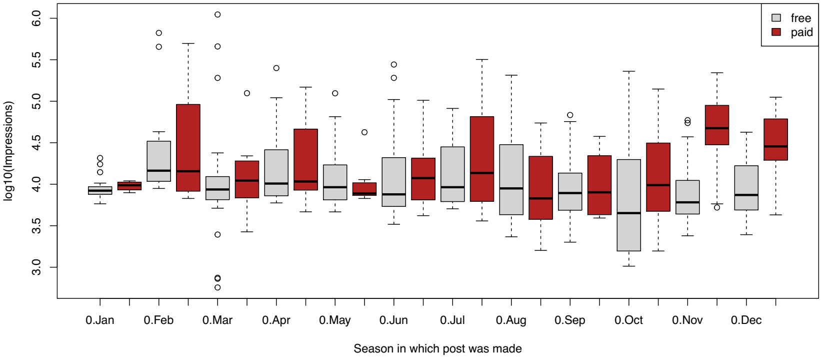



Comments?

# *Facebook data - other uses of formula*

How about?

```
boxplot(log10(Impressions) ~ Paid + cut(Post.Month, 4,
                                     labels = c("Winter", "Spring",
                                                "Summer", "Fall")),
        xlab = "Season in which post was made",
        data = facebook, col = rep(c("lightgrey", "firebrick"), 4))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

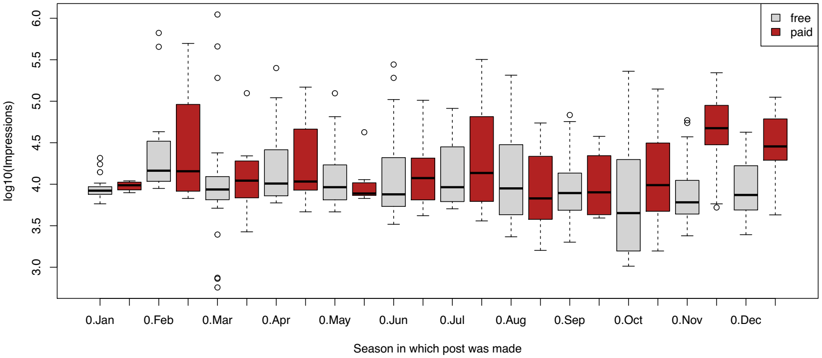



Comments? Huh? What happened here?

# *Facebook data - other uses of formula*

How about?

```
boxplot(log10(Impressions) ~ Paid + cut(Post.Month, 4,
                                        labels = c("Winter", "Spring",
                                                   "Summer", "Fall")),
        xlab = "Season in which post was made",
        data = facebook, col = rep(c("lightgrey", "firebrick"), 4))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```



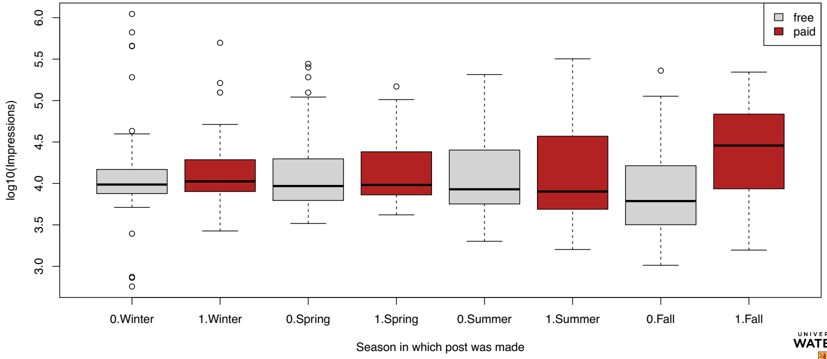

Comments? Huh? What happened here? A bug?

UNIVERSITY OF WATERLOO

# *Facebook data - other uses of formula*

How about?

```r
facebook2 <- data.frame(facebook,
                        season = cut(facebook$Post.Month, 4,
                                     labels = c("Winter", "Spring",
                                                "Summer", "Fall")))
boxplot(log10(Impressions) ~ Paid + season,
        xlab = "Season in which post was made",
        data = facebook2, col = rep(c("lightgrey", "firebrick"), 4))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

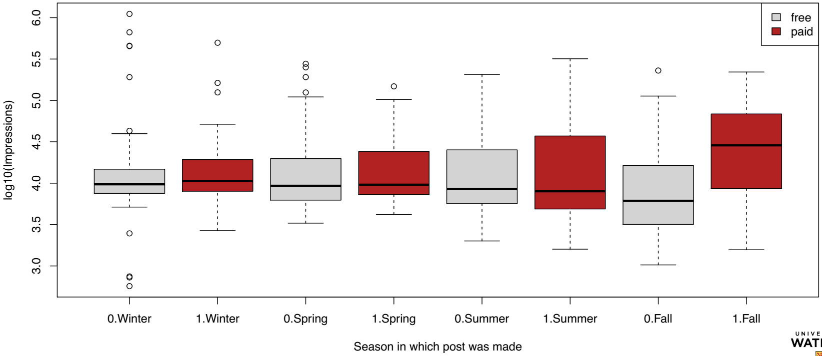



Comments?

# *Facebook data - other uses of formula*

How about?

```r
facebook2 <- data.frame(facebook,
                        season = cut(facebook$Post.Month, 4,
                                     labels = c("Winter", "Spring",
                                                "Summer", "Fall")))
boxplot(log10(Impressions) ~ Paid + season,
        xlab = "Season in which post was made",
        data = facebook2, col = rep(c("lightgrey", "firebrick"), 4))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

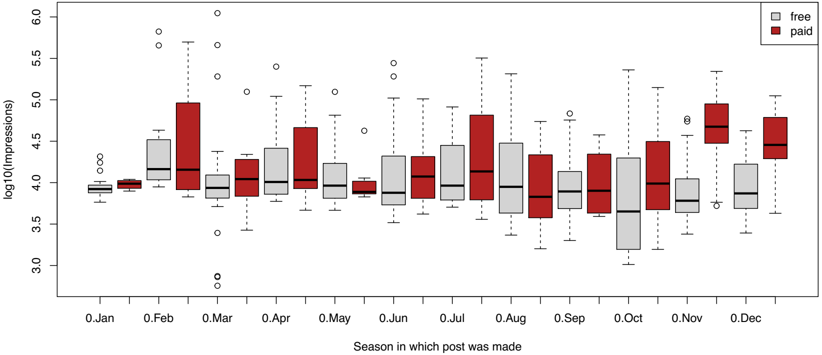



Comments?

# *Facebook data - other uses of formula*

Or how about?

```
boxplot(log10(Impressions) ~ Paid + cut(Post.Month, 12,
                                        labels = month.abb),
        xlab = "Season in which post was made",
        data = facebook, col = rep(c("lightgrey", "firebrick"), 6))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

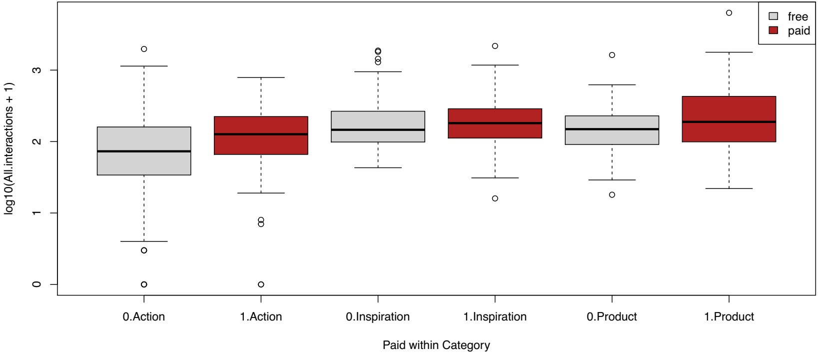



Comments?

# *Facebook data - other uses of formula*

Change response.

```
boxplot(log10(All.interactions + 1) ~ Paid + Category,
        xlab = "Paid within Category",
        data = facebook, col = rep(c("lightgrey", "firebrick"), length(facebook$Category)))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```




Comments?

# *Facebook data - other uses of formula*

Change response. A slightly different fitted model

```r
fit4 <- lm(log10(All.interactions + 1) ~ Paid * Category,  data = facebook)
summary(fit4)
```

```
##
## Call:
## lm(formula = log10(All.interactions + 1) ~ Paid * Category, data = facebook)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.01793 -0.23100  0.02586  0.27246  1.51761
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.77795    0.03941  45.111  < 2e-16 ***
## Paid                         0.23998    0.07224   3.322  0.00096 ***
## CategoryInspiration          0.46570    0.06040   7.711 6.96e-14 ***
## CategoryProduct              0.37776    0.06302   5.994 3.95e-09 ***
## Paid:CategoryInspiration    -0.25185    0.11299  -2.229  0.02627 *
## Paid:CategoryProduct        -0.04374    0.12234  -0.358  0.72083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4843 on 493 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.1524, Adjusted R-squared:  0.1438
## F-statistic: 17.72 on 5 and 493 DF,  p-value: 3.639e-16
```

Comments?

# *Facebook data - other uses of formula*

Change response. A slightly different fitted model

```r
fit4$coefficients
```

```
##              (Intercept)                     Paid      CategoryInspiration
##               1.77795481               0.23997989               0.46569543
##          CategoryProduct Paid:CategoryInspiration     Paid:CategoryProduct
##               0.37776394              -0.25185230              -0.04374152
```

```r
head(model.matrix(fit4))
```

```
##   (Intercept) Paid CategoryInspiration CategoryProduct Paid:CategoryInspiration
## 1           1    0                   0               1                        0
## 2           1    0                   0               1                        0
## 3           1    0                   1               0                        0
## 4           1    1                   0               1                        0
## 5           1    0                   0               1                        0
## 6           1    0                   0               1                        0
##   Paid:CategoryProduct
## 1                    0
## 2                    0
## 3                    0
## 4                    1
## 5                    0
## 6                    0
```

Comments?

# *Facebook data - other uses of formula*

Change explanatory variates to two factors

```r
fit5 <- lm(log10(All.interactions + 1) ~ Type * Category,  data = facebook)
summary(fit5)
```

```
##
## Call:
## lm(formula = log10(All.interactions + 1) ~ Type * Category, data = facebook)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83783 -0.23716  0.01508  0.25688  1.60199
##
## Coefficients: (2 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     1.73278    0.10895  15.904   <2e-16 ***
## TypePhoto                       0.10504    0.11469   0.916   0.3602
## TypeStatus                      0.36213    0.30167   1.200   0.2306
## TypeVideo                       0.65020    0.21397   3.039   0.0025 **
## CategoryInspiration             0.20172    0.49927   0.404   0.6864
## CategoryProduct                -0.03381    0.49927  -0.068   0.9460
## TypePhoto:CategoryInspiration   0.20383    0.50213   0.406   0.6850
## TypeStatus:CategoryInspiration -0.09280    0.62270  -0.149   0.8816
## TypeVideo:CategoryInspiration        NA         NA      NA       NA
## TypePhoto:CategoryProduct       0.39575    0.50315   0.787   0.4319
## TypeStatus:CategoryProduct      0.16441    0.57849   0.284   0.7764
## TypeVideo:CategoryProduct            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4872 on 490 degrees of freedom
## Multiple R-squared:  0.1473, Adjusted R-squared:  0.1316
## F-statistic: 9.405 on 9 and 490 DF,  p-value: 3.024e-13
```

Comments?

# *Facebook data - other uses of formula*

Just the coefficients.

```
fit5$coefficients
```

```
##                     (Intercept)                       TypePhoto
##                      1.73278345                      0.10504178
##                      TypeStatus                       TypeVideo
##                      0.36213204                      0.65020444
##             CategoryInspiration                 CategoryProduct
##                      0.20171500                     -0.03381344
##   TypePhoto:CategoryInspiration TypeStatus:CategoryInspiration
##                      0.20382942                     -0.09279854
##   TypeVideo:CategoryInspiration      TypePhoto:CategoryProduct
##                              NA                      0.39574802
##      TypeStatus:CategoryProduct      TypeVideo:CategoryProduct
##                      0.16440892                              NA
```

# *Facebook data - other uses of formula*

And now the corresponding model matrix.

```
head(model.matrix(fit5))
```

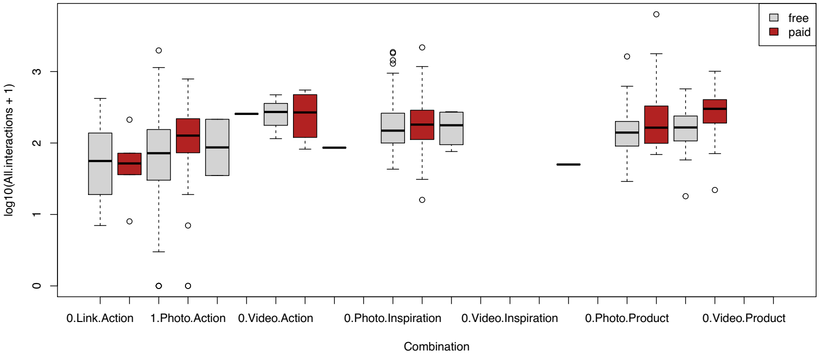
```
##   (Intercept) TypePhoto TypeStatus TypeVideo CategoryInspiration
## 1           1         1          0         0                   0
## 2           1         0          1         0                   0
## 3           1         1          0         0                   1
## 4           1         1          0         0                   0
## 5           1         1          0         0                   0
## 6           1         0          1         0                   0
##   CategoryProduct TypePhoto:CategoryInspiration TypeStatus:CategoryInspiration
## 1               1                             0                              0
## 2               1                             0                              0
## 3               0                             1                              0
## 4               1                             0                              0
## 5               1                             0                              0
## 6               1                             0                              0
##   TypeVideo:CategoryInspiration TypePhoto:CategoryProduct
## 1                             0                         1
## 2                             0                         0
## 3                             0                         0
## 4                             0                         1
## 5                             0                         1
## 6                             0                         0
##   TypeStatus:CategoryProduct TypeVideo:CategoryProduct
## 1                          0                         0
## 2                          1                         0
## 3                          0                         0
## 4                          0                         0
## 5                          0                         0
## 6                          1                         0
```

What does the intercept term represent?

# *Facebook data - other uses of formula*

Or how about?

```
boxplot(log10(All.interactions + 1) ~ Paid + Type + Category,
        xlab = "Combination",
        data = facebook, col = rep(c("lightgrey", "firebrick"), length(facebook$Type) * ler
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

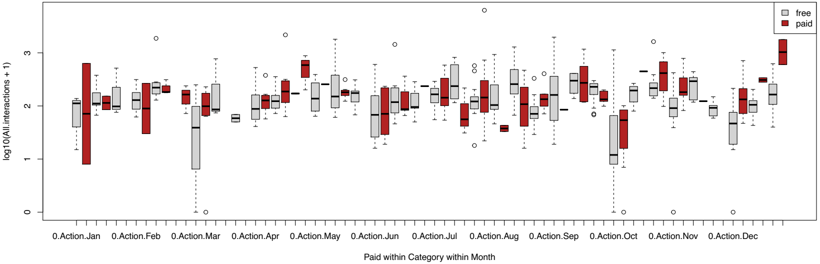



Comments?

# *Facebook data - other uses of formula*

Or

```
boxplot(log10(All.interactions + 1) ~ Paid + Category + cut(Post.Month, 12, labels = month.abb),
        xlab = "Paid within Category within Month",
        data = facebook, col = rep(c("lightgrey", "firebrick"), 12 * length(facebook$Category)))
legend("topright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

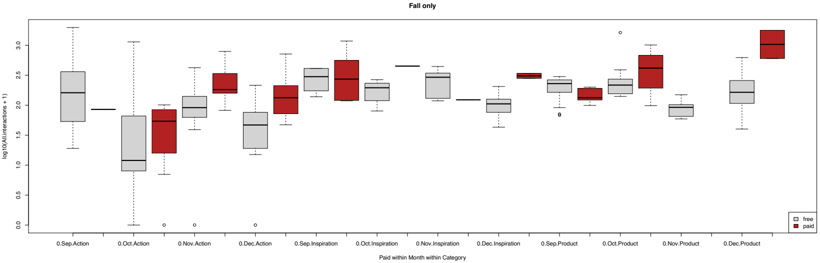



Comments?

# *Facebook data - other uses of formula*

## Focus only on Fall

```
Fall <- facebook$Post.Month %in% 9:12
with(facebook[Fall,], {
    boxplot(log10(All.interactions + 1) ~ Paid + cut(Post.Month, 4, labels = month.abb[9:12])+ Category ,
            xlab = "Paid within Month within Category", main = "Fall only",
            col = rep(c("lightgrey", "firebrick"), 4 * length(Category)))
    legend("bottomright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
}
)
```



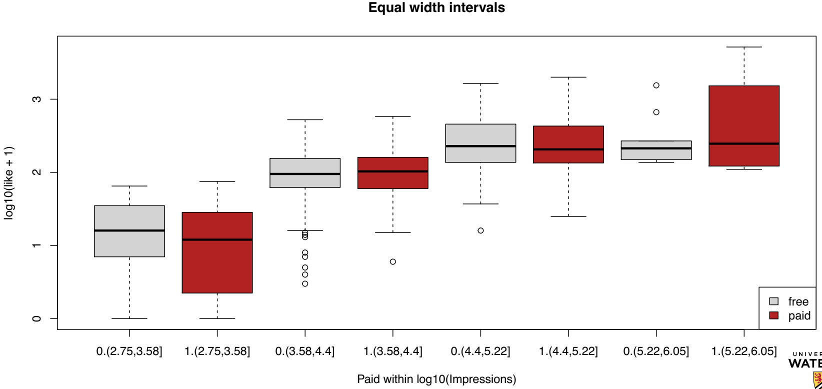

Comments?

## *What about a continuous explanatory variate?*

Could use cut() on the continuous (ratio-scaled) variate to turn it into a categorical and proceed as before.

For example equal width intervals:

```
boxplot(log10(like +1) ~ Paid + cut(log10(Impressions), 4),
        xlab = "Paid within log10(Impressions)",
        data = facebook, main = "Equal width intervals",
        col = rep(c("lightgrey", "firebrick"), 2))
legend("bottomright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```

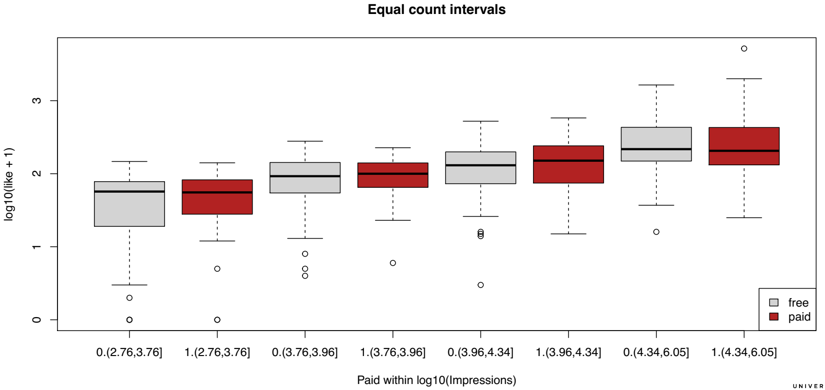



Comments?

# *What about a continuous explanatory variate?*

Or perhaps four intervals of equal numbers

```
boxplot(log10(like +1) ~ Paid + cut(log10(Impressions),
                                    breaks = quantile(log10(Impressions))),
        xlab = "Paid within log10(Impressions)",
        data = facebook, main = "Equal count intervals",
        col = rep(c("lightgrey", "firebrick"), 2))
legend("bottomright", legend = c("free", "paid"), fill = c("lightgrey", "firebrick"))
```




Comments?

## *What about a continuous explanatory variate?*

Alternatively, we could build a (perhaps complicated) linear model, say modelling the mean of response $Y$ as a polynomial of explanatory variate $x$:

$$\mu(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p$$

(or as any other linear (in the coefficients) model).

Such models can be fitted by least-squares.

Unfortunately, these models require a parametric form (e.g. a polynomial) be specified that will fit the data **everywhere** (i.e. globally for all $x$).

Alternatively, we could try fitting many simple functions of $x$ **locally**, different at every value of $x$. Connecting the fitted values together produces an estimated $\mu(x)$

UNIVERSITY OF WATERLOO

## *What about a continuous explanatory variate?*

For example, while we might not be willing to have one line fit all points, we might be willing to have *different lines* fitted in separate (and contiguous) regions of $x$. That is we could fit lines **locally** within each region of $x$.

We can fit locally by using **weighted least squares** which minimizes

$$\sum_{i=1}^{n} w_i(x)\left(y_i - \mu(x_i)\right)^2$$

where $w_i(x)$ depends on the location $x$ where we are fitting $\mu(x)$. We fit $\mu(x)$ for every $x$ on in the range of the data.

We could also make the weight function $w_i(x)$ to be 1 for those $x_i$ near $x$ and 0 for those far away. In this way, the weights determine the $x_i$ values that contribute to fitting $\mu(x)$ and those which do not.

For example, for the $i$th observation $x_i$ when fitting at any point $x$ we could have

$$w_i(x) = K\left(\frac{x_i - x}{a}\right)$$

for some $a > 0$ and function $K(t) \geq 0$ that is maximal at $(t = 0)$ and decreasing with $|t|$.

The constant $a$ controls the size of the region (i.e. which $x_i$s) that will be used to fit at $x$. The smaller these regions were, the less structure we would be imposing on the underlying function $\mu(x)$.

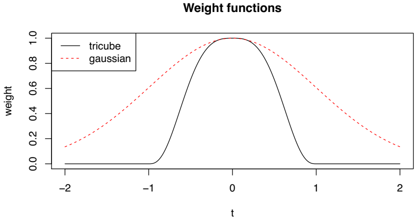
# *Locally weighted sum of squares fitting – `loess()`*

In R there is a function called `loess` that fits a locally weighted sum of squares estimate that pays a little more attention to some of these problems.

```
loess(formula, data, ..., span = 0.75, ...)
```

As its default weight function loess uses Tukey's tri-cube weight

$$K(t) = \begin{cases} (1 - |t|^3)^3 & t \in [-1, 1] \\ 0 & \text{otherwise.} \end{cases}$$




This is generally more resistant to outlying $x_i$s than, say, a Gaussian weight function (i.e. $K(W) = \phi(w)$ the N(0,1) density).

N.B. `loess` is not restricted to fitting local lines. It can fit any degree polynomial locally (though typically only degree 1 or 2 is used in practice).

# *Locally weighted sum of squares fitting – `loess()`*

```
fit2 <- loess(log10(like + 1) ~ log10(Impressions), data=facebook , span = 0.3)
plot(log10(facebook$Impressions), log10(facebook$like + 1),
     col="grey50", pch=19, main = "Loess smooth")
# Get the values predicted by the loess
xmin <- min(facebook$Impressions)
xmax <- max(facebook$Impressions)
x <- seq(xmin, xmax, length.out = 200)
pred <- predict(fit2, newdata = data.frame(Impressions = x))
lines(log10(x),pred, lwd=3, col="steelblue")
```